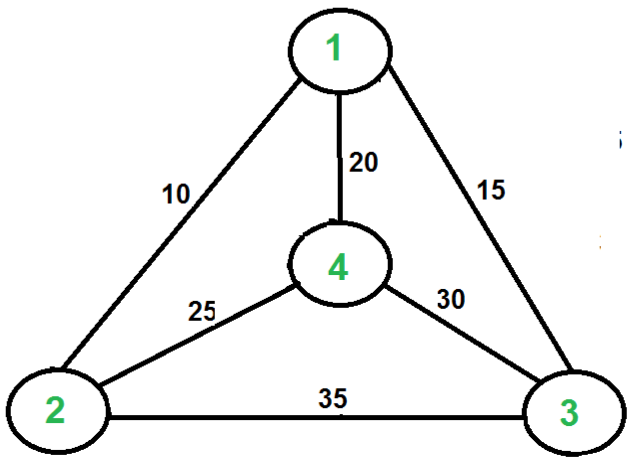
Travelling Salesman Problem (TSP)

Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point. Note the difference between Hamiltonian Cycle and TSP. The Hamiltonian cycle problem is to find if there exist a tour that visits every city exactly once. Here we know that Hamiltonian Tour exists (because the graph is complete) and in fact many such tours exist, the problem is to find a minimum weight Hamiltonian Cycle.



Source : http://www.geeksforgeeks.org/travelling-salesman-problem-set-1/

For example, consider the graph shown in the top figure. A TSP tour in the graph is 1-2-4-3-1. The cost of the tour is 10+25+30+15 which is 80.

The problem is a famous NP hard problem. There is no polynomial time know solution for this problem.

For our implementation we have assumed that nodes are named from 0 to n-1 if there are n nodes(cities). Also the implementation has assumed that 0 is the starting state. But this assumption has no major impact as we can change the starting city and calculate the shortest path. Our major concern is how would we implement the TSP using greedy and dynamic approach. Finding the shortest path from 0 node around the graph is same as finding the shortest path in graph from every node and then finding the minimum path. We try to solve the subproblem using dynamic and greedy.

The two approaches in discussion are dynamic and greedy as follows:

Greedy is the most naive approach that we can consider. In Greedy we start with node 0 and find the shortest path from node 0 to all other nodes. The path with the shortest route is selected. After that we move along the path and from the current node choose another path that is the shortest. But we make sure we don't traverse back to the nodes/cities that we have already visited. Since this is the property of TSP.

The dynamic approach is the interesting one. Consider a Hamiltonian graph. We are given the weights of each path in the graph. If we construct all paths from source city 0 among our graph then we are obviously point out the minimum path. Constructing these paths follow the 2 properties of dynamic programming

1. Overlapping Subproblem
2. Optimal Substructure

If we consider a node x in the graph. Then many paths exists from the source node 0. Since many paths exists the solution of finding the shortest path to x would be used by all these paths. Also by solving the minimum path till node x we can use the solution to find the optimal solution of path from node 0 to 0 covering all nodes in the graph.

Recursive Dynamic Solution : (Held-Karp Algorithm)

Number the cities as 0, 2, . . . , n-1 and assume we start at city 0, and the distance between city i and city j is d_{ij}. Now consider subsets of $S \subseteq \{1, . . . , n\text{-}1\}$ of cities where $c \in S$

Let D(S, c) be the minimum distance, starting at city 0, visiting all cities in S and then finishing at city c.

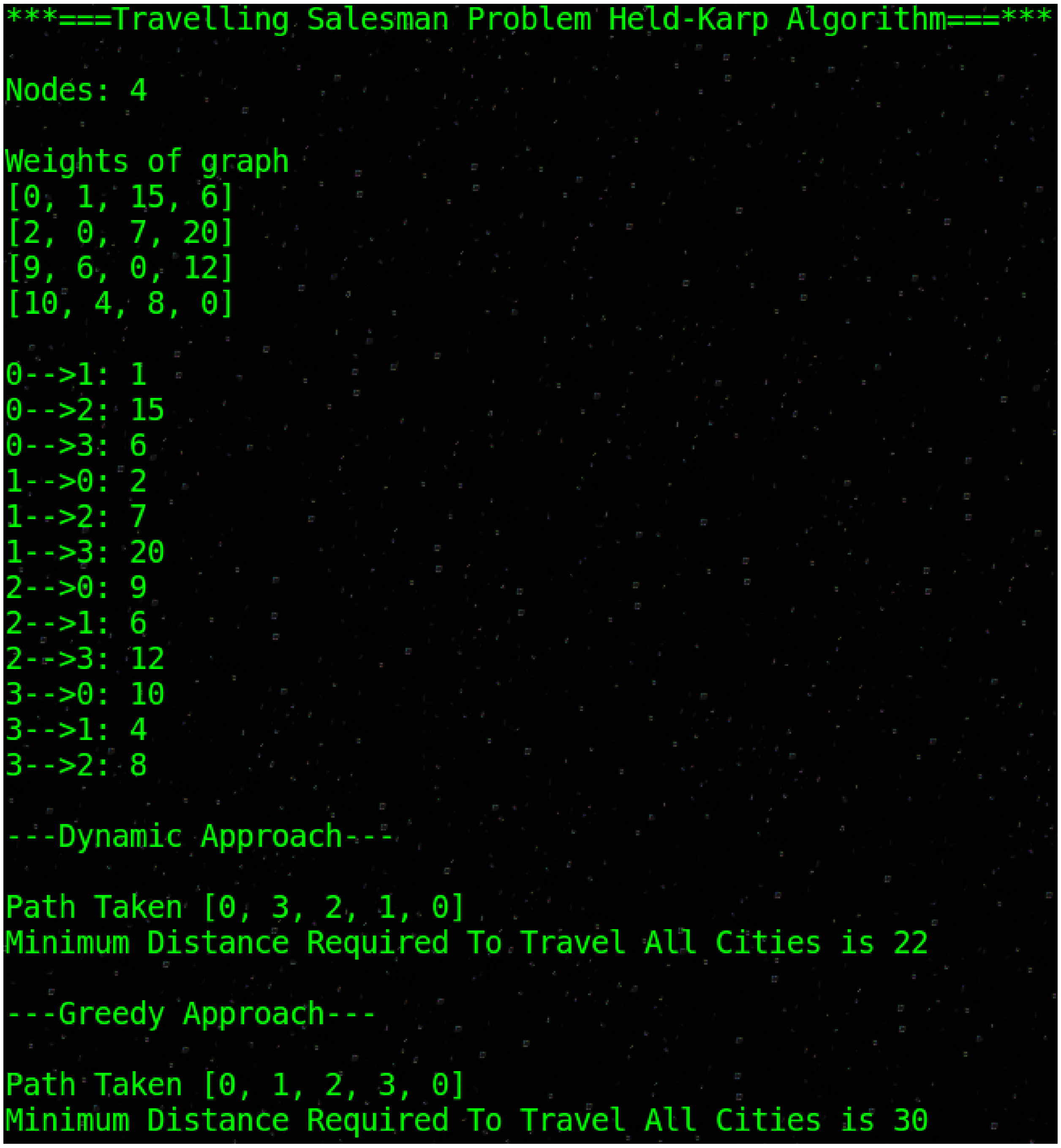
1. If $S = \{c\}$, then $D(S, c) = d_{0,c}$. Otherwise: $D(S, c) = \min_{x \in S - c} (D(S - c, x) + d_{x,c})$
2. The minimum distance for a complete tour of all cities is $M = \min_{c \in \{1,\dots,n\text{-}1\}} (D(\{1, . . . , n\text{-}1\}, c) + d_{c,0})$

Complexity	Dynamic	Greedy
Time	2^n * n^2	n^2
Space	2^n * n^2	n

We challenged ourselves by implementing the TSP dynamic programming problem and TSP greedy algorithm :

The code is implemented in tsp.py

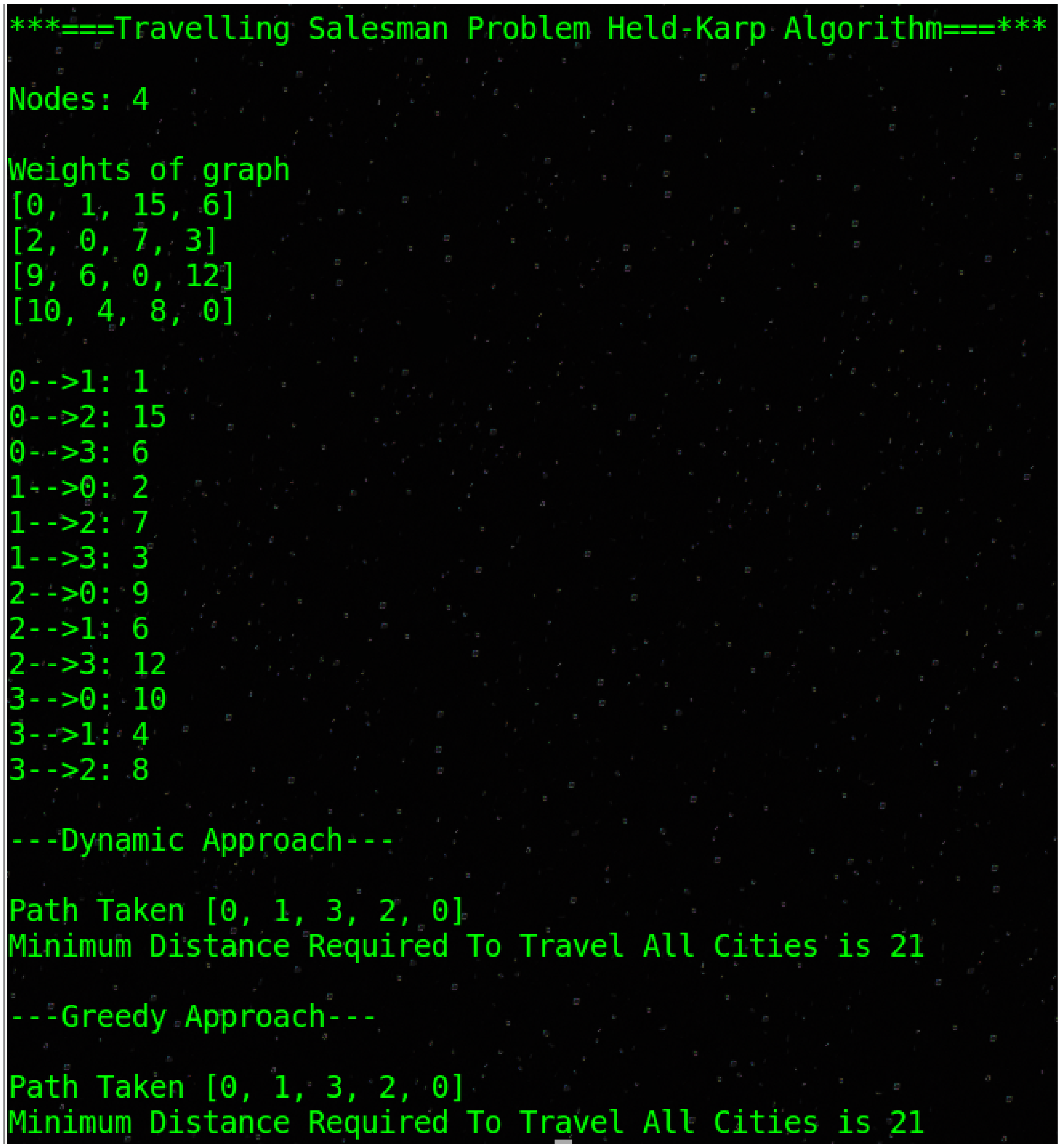
Below are 2 run examples of TSP



```
***===Travelling Salesman Problem Held-Karp Algorithm===***

Nodes: 4

Weights of graph
[0, 1, 15, 6]
[2, 0, 7, 20]
[9, 6, 0, 12]
[10, 4, 8, 0]

0-->1: 1
0-->2: 15
0-->3: 6
1-->0: 2
1-->2: 7
1-->3: 20
2-->0: 9
2-->1: 6
2-->3: 12
3-->0: 10
3-->1: 4
3-->2: 8

---Dynamic Approach---

Path Taken [0, 3, 2, 1, 0]
Minimum Distance Required To Travel All Cities is 22

---Greedy Approach---

Path Taken [0, 1, 2, 3, 0]
Minimum Distance Required To Travel All Cities is 30
```

The Weight of graph is a tabular representation of weights of path in the graph.

In this case it is easily seen that greedy algorithm by making a greedy choice does not make an optimal solution. Whereas the dynamic approach does produce an optimal solution



```
***===Travelling Salesman Problem Held-Karp Algorithm===***

Nodes: 4

Weights of graph
[0, 1, 15, 6]
[2, 0, 7, 3]
[9, 6, 0, 12]
[10, 4, 8, 0]

0-->1: 1
0-->2: 15
0-->3: 6
1-->0: 2
1-->2: 7
1-->3: 3
2-->0: 9
2-->1: 6
2-->3: 12
3-->0: 10
3-->1: 4
3-->2: 8

---Dynamic Approach---

Path Taken [0, 1, 3, 2, 0]
Minimum Distance Required To Travel All Cities is 21

---Greedy Approach---

Path Taken [0, 1, 3, 2, 0]
Minimum Distance Required To Travel All Cities is 21
```

The Weight of graph is a tabular representation of weights of path in the graph.
An example where greedy and dynamic both produce an optimal solution.

Conclusion

TSP is still a NP Hard problem. Dynamic Programming produces a solution but the complexity is exponential and given a humongous graph which generally is the case. It will be quite a problem when calculating the minimum path. Also it has to be noted that we have solved the TSP by assuming node 0 as the origin. We still need to run the same algorithm/code again for all remaining n-1 nodes and then print the minimum path. This will have infeasible exponential complexity. Whereas greedy/naive algorithm will give a very poor answer if the graph size is increased. From here we can build on a less complexity algorithm with constrained alpha-approximation taking the idea of dynamic programming.